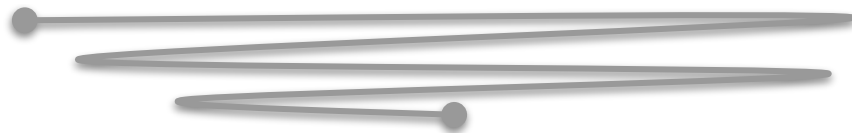


Laboratorio de Datos



Introducción a Python // Parte 01



2do. Cuatrimestre
2023



DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA



Contenido

- + Python
- + Pythontutor
- + Tipos de datos básicos - operaciones
- + Tipos de datos que contienen otros datos.
- + Condicional y ciclos

Laboratorio de Datos

Introducción a Python - Parte 1

... por Manuela Cerdeiro (y modificaciones de P. Turjanski)

Python

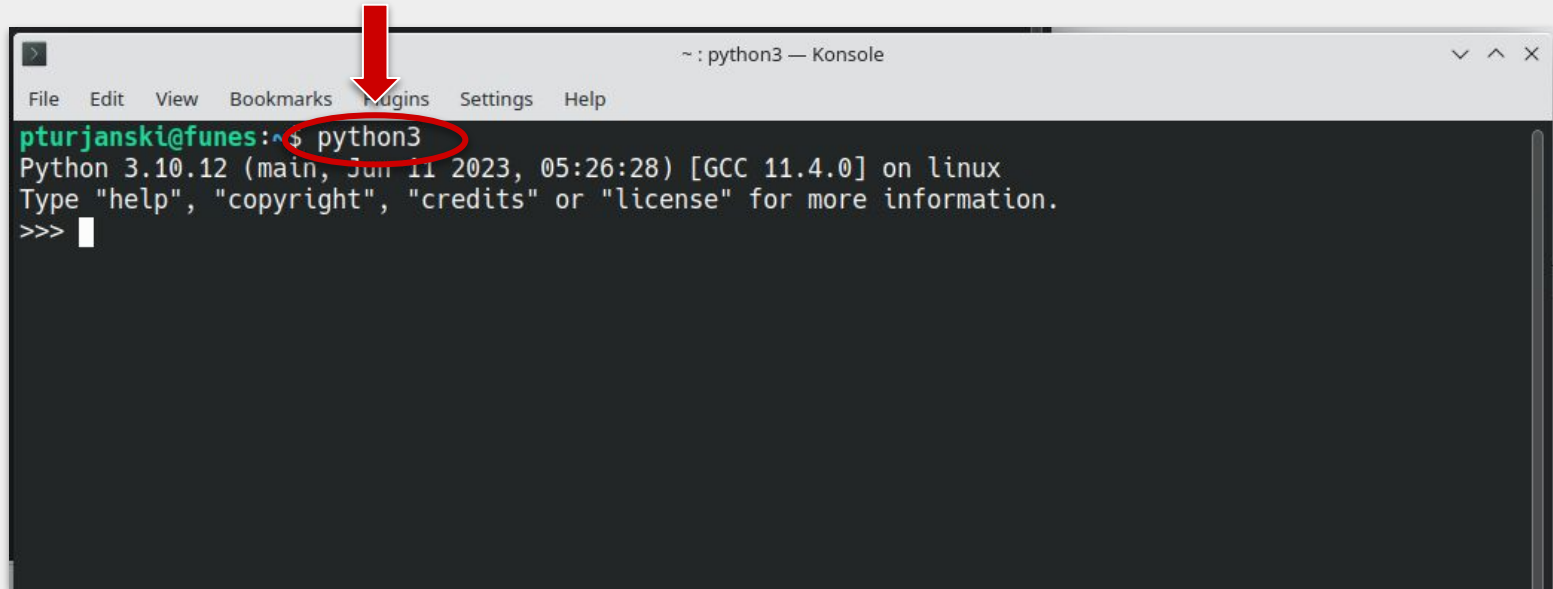


Python

- + Código abierto (open source)
- + Multiplataforma (sirve en linux, ios, windows...)
- + Tiene muchas herramientas para ciencias de datos (y para muchas otras disciplinas)
- + Alto nivel, sencillo de aprender
- + Mucha, mucha, mucha gente lo usa (facilita la consulta)
- + Algunxs ya lo usaron en Algo 1/IP

Acceder a python a través de la terminal ...

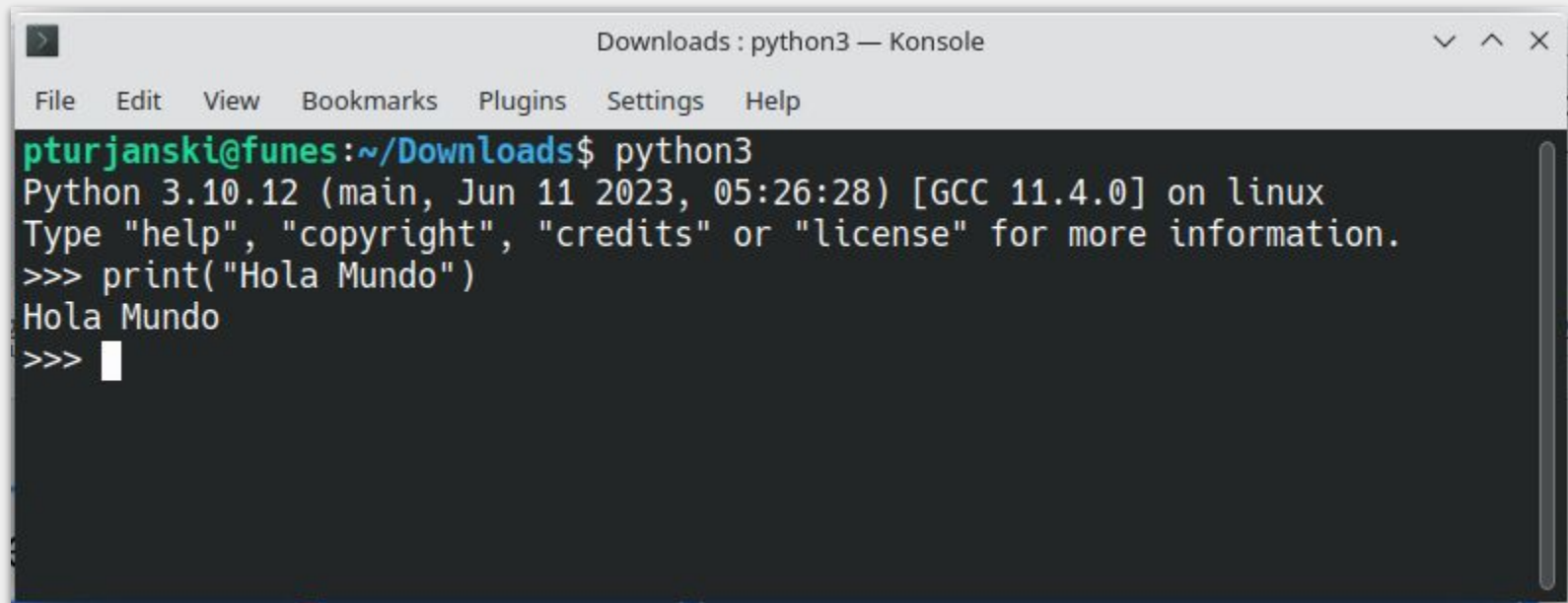
Varias maneras de utilizarlo. Comenzamos con interacción mediante intérprete o consola.



A screenshot of a terminal window titled "python3 — Konsole". The window has a menu bar with "File", "Edit", "View", "Bookmarks", "Plugins", "Settings", and "Help". A red arrow points to the "Plugins" menu. The terminal shows the prompt "pturjanski@funes:~\$" followed by the command "python3", which is circled in red. The output of the command is "Python 3.10.12 (main, Jun 11 2023, 05:26:28) [GCC 11.4.0] on linux" and "Type 'help', 'copyright', 'credits' or 'license' for more information." followed by a new prompt ">>>".

```
pturjanski@funes:~$ python3
Python 3.10.12 (main, Jun 11 2023, 05:26:28) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Primer ejercicio -> Imprimir "Hola Mundo" ...

A screenshot of a terminal window titled "Downloads : python3 — Konsole". The window has a menu bar with "File", "Edit", "View", "Bookmarks", "Plugins", "Settings", and "Help". The terminal content shows a user prompt "pturjanski@funes:~/Downloads\$" followed by the command "python3". The output displays the Python version "Python 3.10.12 (main, Jun 11 2023, 05:26:28) [GCC 11.4.0] on linux" and instructions to type "help", "copyright", "credits", or "license" for more information. The user then enters the command ">>> print('Hola Mundo')", and the terminal outputs "Hola Mundo". The prompt ">>>" is followed by a white cursor block.

```
pturjanski@funes:~/Downloads$ python3
Python 3.10.12 (main, Jun 11 2023, 05:26:28) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hola Mundo")
Hola Mundo
>>> 
```

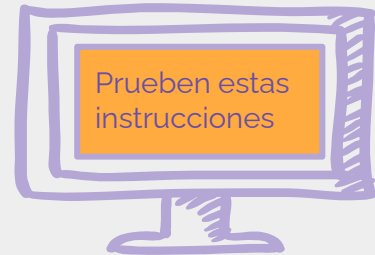
Asignación de variables

```
valorDolarOficial = 280
```

Esta instrucción crea una variable llamada `valorDolarOficial` y le asigna el valor 280. Si queremos modificar su valor podemos hacer lo siguiente:

```
valorDolarOficial = 296 # ahora a vale 296
```

```
# El contenido posterior a un símbolo numeral (#)  
# es un comentario y Python no lo ejecuta
```



Tipos de datos básicos

Tipos de datos básicos

- + `int` : Representan números enteros. Ejemplo: 1, 2, -5, 102978
- + `float` : Representan números reales. Ejemplo: 56.842
- + `bool` : Representan valores booleanos. Ejemplo: True/False (1/0)

Operaciones - Números enteros

Operaciones con números enteros

`280 + 16` `# Este valor no se lo asigna a ninguna variable`

`base = 3`

`altura = 4`

`supRectangulo = base*altura` `# Calcula la superficie del rectángulo`

`supTriangulo = (base*altura)//2` `# El símbolo // es para la división entera`

`cantAlumnes = 53` `# Cantidad de alumnos en el curso`

`alumnesSinGrupo = cantAlumnes%3` `# El símbolo % calcula el resto de la división entera`

`unkByte = 2**10` `# El doble ** es la potenciación.`
 `# 2**10=1024 (1024 bytes equivalen a 1 kb)`

Operaciones - Valores Bool

Operaciones con valores bool

- + and
- + or
- + not

Representan la conjunción, disyunción y negación, respectivamente.

```
soyProfesor = True
```

```
habloFrances = False
```

```
conjuncion = soyProfesor and habloFrances
```

```
disyuncion = soyProfesor or habloFrances
```

```
# soyProfesor → habloFrances
```

```
implicacion = (not soyProfesor) or habloFrances
```

Operaciones de int/float a bool

Al realizar comparaciones obtenemos valores de verdad (bool):

```
dolarCompra = 281
```

```
dolarVenta  = 294
```

```
dolarCompra > dolarVenta # Va a dar False
```

```
dolarCompra <= dolarVenta # Va a dar True
```

```
dolarCompra == dolarVenta # Este es el símbolo para ver si son iguales
```

```
dolarCompra != dolarVenta # Este es el símbolo para ver si son distintos
```

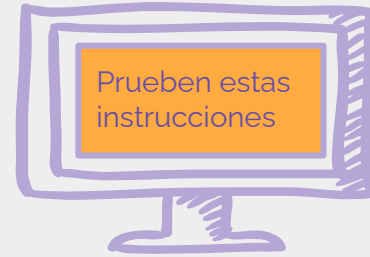
Operaciones de int/float a bool

Al realizar comparaciones obtenemos valores de verdad (bool):

`a = 8`

`b = 15`

`x = 2`



Ejercicio. ¿Qué valores de verdad obtienen?

- i) `(x < a) and (10*x >= b) and not (x == a - b)`
- ii) `(x < a) or (x > b)`

Datos compuestos

- + Cadenas de caracteres
- + Listas
- + Tuplas
- + Conjuntos

Cadenas de caracteres - *Strings*

Son la forma de manipular texto.

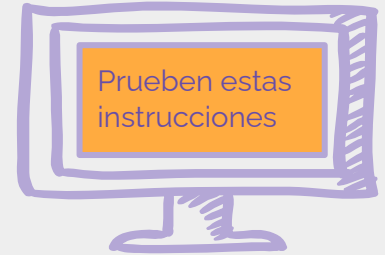
Se construyen con comillas simples o dobles.

```
fraseDelDia = 'Hoy es lunes'
```

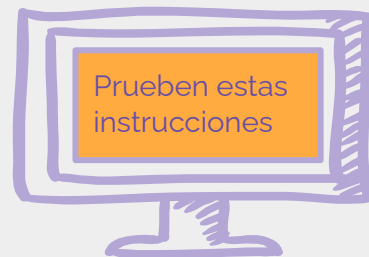
```
fraseComienzo = 'empiezo labo de datos'
```

```
fraseFinal = fraseDelDia + fraseComienzo # Queda feo
```

```
fraseFinal = fraseDelDia + ', ' + fraseComienzo # Así mejor
```



Cadenas de caracteres - *Strings*



```
fraseDelDia = 'Hoy es lunes'
```

```
fraseDelDia[4]    # Corresponde a 'e' (comienza a contar en 0)
```

```
fraseDelDia[4:8]  # Corresponde a 'es l' (excluye la posición 8)
```

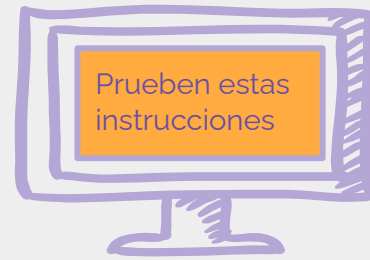
```
fraseDelDia[4:]   # Corresponde a 'es lunes'
```

```
fraseDelDia[:5]   # Corresponde a 'Hoy e' (excluye la posición 5)
```

Cadenas de caracteres - *Strings*

```
fraseDelDia = 'Hoy es lunes'
```

```
fraseDelDia[4]= 'E'
```



```
Downloads : python3 — Konsole
File Edit View Bookmarks Plugins Settings Help
>>> fraseDelDia = 'Hoy es lunes'
>>> fraseDelDia[4]= 'E'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>>
>>>
>>>
>>>
>>>
>>>
```

Cadenas de caracteres - *Strings*

```
a = 'Hello' + 'World'    # Concatenación: 'HelloWorld'
b = 'Say ' + a           # 'Say HelloWorld'
s = 'Hello'
len(s)                   # Longitud (5)
t = 'e' in s             # Chequea si pertenece : True
f = 'x' in s             # Chequea si pertenece : False
g = 'hi' not in s        # Chequea si NO pertenece: True
rep = s * 5              # 'HelloHelloHelloHelloHello'
l = s.lower()            # 'hello'
u = s.upper()            # 'HELLO'
```

Print

La función print imprime en pantalla

```
print('Hola Mundo')           # Imprime 'Hola Mundo'
```

```
a = 'Hoy es muuuuy lunes'
```

```
print(a)          # Imprime 'Hoy es muuuuy lunes'
```

```
numeroDeClase = 1
```

```
print(numeroDeClase)      # También imprime cosas que
                          # no son strings
```

[illegible]

Ejercicio

Escribir un programa que decida si una palabra de longitud exactamente **5** es palíndromo. Ej. "NADAN" es palíndromo. "JUJUY" no lo es.



Pythontutor

Python Tutor - <https://pythontutor.com/>

Python Tutor: Visualize code in [Python](#), [JavaScript](#), [C](#), [C++](#), and [Java](#)

Python 3.6
[known limitations](#)

```
1 palabra = 'JUJUY'  
→ 2 print("Es palindromo",palabra[0]==palabra[4] and palabra[1]==palabra[3])
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Done running (2 steps)

Print output (drag lower right corner to resize)

Es palindromo False

Frames

Objects

Global frame

palabra "JUJUY"

Listas

Pueden contener todo tipo de variables.

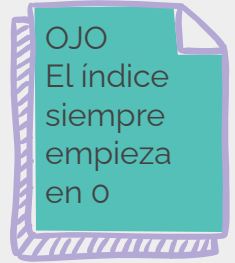
Las listas se usan mucho. Se construyen con corchetes [].

```
lista_numeros = [10, 43, 22, 5, 63, 101, -5, 3]
```

```
lista_nombres = ['Julia', 'Luciana', 'Manuel']
```

```
lista_vacia = []
```

Listas



```
lista_numeros = [10, 43, 22, 5, 63, 101, -5, 3]
```

```
lista_numeros[0]      # Accede al 1er elemento de la lista: 10
```

```
lista_numeros[3]      # Accede al 4to elemento de la lista: 5
```

```
lista_numeros[7]      # Accede al 8vo elemento de la lista: 3
```

```
lista_numeros[1:5]    # Rebanada/slice: [43, 22, 5, 63]
```

```
len(lista_numeros)    # Tamaño de la lista: 8
```

Listas


```
lista_nombres = ['Julia', 'Luciana', 'Manuel']
```

```
lista_nombres.append('Pedro') # Agrega un elemento ('Pedro') al final
```

```
lista_nombres[1] = 'Ana'
```

Listas

```
lista_nombres = ['Julia', 'Luciana', 'Manuel']
```



Las listas
son
mutables


```
lista_nombres.append('Pedro') # Agrega un elemento ('Pedro') al final
```

```
lista_nombres[1] = 'Ana' # Cambia el segundo elemento
```



Listas

```
lista_nombres = ['Julia', 'Luciana', 'Manuel']
```



Las listas
son
mutables

```
lista_nombres.append('Pedro') # Agrega un elemento ('Pedro') al final
```

```
lista_nombres[1] = 'Ana' # Cambia el segundo elemento
```

```
'Manuel' in lista_nombres # Chequea si Manuel está en la  
                           # lista: True
```

```
'Julia' not in lista_num # Chequea si Julia NO está en la lista
```

```
lista_nombres.remove('Julia') # Remueve a Julia de la lista
```

```
lista_numeros.sort() # Ordena la lista
```

Listas

```
frutas = 'Frambuesa,Manzana,Naranja,Mandarina,Banana,Sandía,Pera'
lista_frutas = frutas.split(',') # separa en las comas
lista_frutas[0]    # Frambuesa
lista_frutas[1]# Manzana
lista_frutas[-1]   # Último elemento: Pera
lista_frutas[-2]   # Anteúltimo elemento: Sandía
lista_frutas.append('Limón')    # Agrega elemento al final
lista_frutas.insert(0, 'Limón') # Insertar al principio
lista_frutas += ['Frutilla', 'Palta'] # Concatena listas
```

Listas

```
lista_frutas = ['Frambuesa', 'Manzana', 'Naranja']
```

```
compras      = ['café', lista_frutas, 'huevos'] # ¡Tiene una lista  
                                                  adentro!
```

```
compras[1]          # ¿Qué devuelve?
```

```
compras[1][1]       # ¿Qué devuelve?
```

```
mandados = [compras, 'cajero', 'pasear al perro']
```

```
mandados[0][1][2]   # ¿Qué devuelve?
```

```
len(mandados)       # ¿Qué devuelve?
```

```
compras in mandados # ¿Qué devuelve?
```

```
lista_frutas in mandados # ¿Qué devuelve?
```

compras

```
['café', lista_frutas, 'huevos']
```

compras[1]

mandados

```
[['café', ['Frambuesa', 'Manzana', 'Naranja'], 'huevos'], 'cajero',  
'pasear al perro']
```

compras[0][1][2]

Matrices como listas de listas

Podemos utilizar listas de listas para definir matrices.
Cada lista representa una fila.

Ejemplos:

$M = [[1, 2], [3, 4]]$

1	2
3	4

$N = [[1, 0, 3], [0, 5, 5], [8, -1, 1], [10, 3, 1]]$

1	0	3
0	5	5
8	-1	1
10	3	1

Matrices como listas de listas

¿Cómo accedemos a los elementos?

`M[0] ?`

`M[0][1] ?`

1	2
3	4

`N[1][2] ?`

`N[?][?] # Quiero acceder a la celda de valor -1`

1	0	3
0	5	5
8	-1	1
10	3	1

Tuplas

Son como vectores. Se arman con paréntesis ().

$a = (2, 4)$

$b = (-1, 2)$

$c = a + b$

$d = (1, 5, 10, 3, 7)$

$a[1]$

$d[9]$

$d[1:5]$

$d[2] = 11$

Tuplas

Son como vectores. Se arman con paréntesis ().

```
a = (2, 4)
```

```
b = (-1, 2)
```

```
c = a + b
```

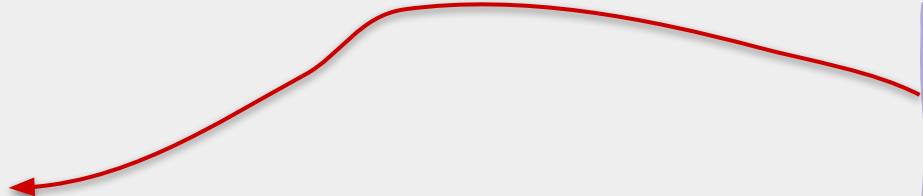
```
d = (1, 5, 10, 3, 7)
```

```
a[1]
```

```
d[9]
```

```
d[1:5]
```

```
d[2] = 11
```



OJO
Las tuplas
son
inmutables

Tuplas

Son como vectores. Se arman con paréntesis ().

```
a = (2, 4)
```

```
b = (-1, 2)
```

```
c = a + b
```

```
d = (1, 5, 10, 3, 7)
```


```
a[1]
```

```
d[9]
```

```
d[1:5]
```

```
d[2] = 11
```

```
d = (1, 5, 11, 3, 7)
```



OJO
Las tuplas
son
inmutables

Conjuntos

Se arman con llaves {}.

```
algunosNaturalesPares = {2, 4}                # Conjunto de valores  
                                                # enteros  
  
citricos = {'Naranja', 'Limón', 'Mandarina'}   # Conjunto de strings  
  
citricos = set(['Naranja', 'Limón', 'Mandarina']) # Conjunto a partir de  
                                                # lista de valores  
  
'Naranja' in citricos                        # ¿Pertenece?  
  
citricos.add('Pomelo')                        # Agrega un elemento  
  
citricos.remove('Naranja')                    # Elimina un elemento  
  
len(citricos)                                # Cardinal
```

Condicional y ciclos

Condicional

Si queremos supeditar la instrucción a una determinada condición, utilizamos el condicional if.

```
if a > 0:
```

```
    print('a es un número positivo!')
```


Condicional Simple

Si queremos supeditar la instrucción a una determinada condición, utilizamos el condicional `if`

```
a = 2
```

```
b = 5
```

```
if a > 0:
```

2 puntitos

nueva línea

4 espacios

```
    print('a es un número positivo!')
```

nueva línea

Condicional Simple

Si queremos supeditar la instrucción a una determinada condición, utilizamos el condicional `if`

```
a = 2
```

```
b = 5
```

```
if a > 0:
```

2 puntitos

nueva línea

4 espacios

```
    print('a es un número positivo!')
```

nueva línea

```
    if a == 0:
```

```
        print('a es cero...')
```

```
if b > a:
```

```
    print('b le ganó a a')
```

Condicional Simple

Ejercicio. Completar este código.

```
a = 2
```

```
b = 4
```

```
c = 6
```

```
if - - -:
```

```
    print('b está entre a y c')
```

```
if - - -:
```

```
    print('b es mayor a c')
```



Condicional Compuesto

Else y elif

```
if condicion1:
    instruccion1
    instruccion2
elif condicion2:
    instruccion3
else:
    instruccion4
```

Ejemplo

```
if a % 4 == 0:
    b = a//4
elif a % 4 == 2:
    c = a//2
else:
    print('a es impar')
```

Condicional Compuesto

Else y elif

case en cpp

Ejemplo

```
if condicion1:  
    instruccion1  
    instruccion2  
elif condicion2:  
    instruccion3  
else:  
    instruccion4
```

condiciones: bool



```
if a % 4 == 0:  
    b = a//4  
elif a % 4 == 2:  
    c = a//2  
else:  
    print('a es impar')
```

Ciclo while

Para utilizar `while` vamos a considerar una condición que puede cumplirse o no (una condición bool, `True/False`). Las instrucciones dentro del ciclo se ejecutarán mientras se satisface la condición.

```
i = 0
```

```
while i < 5:
```

```
    print(i)
```

```
    i += 1    # incrementa la variable i en 1
```

Ciclo while


Para utilizar while vamos a considerar una condición que puede cumplirse o no (una condición bool, True/False). Las instrucciones dentro del ciclo se ejecutarán mientras se satisface la condición.

```
i = 0
```

```
while i < 5:
```

```
    print(i)
```

```
    i += 1    # incrementa la variable i en 1
```



guarda del while
donde figura la condición

A purple arrow points from this text box to the condition 'i < 5' in the 'while' statement above.



algo debe cambiar para que
en algún momento
deje de cumplirse la guarda

A purple arrow points from this text box to the increment statement 'i += 1' in the loop body above.

Ejercicio

Escribir un programa que imprima en pantalla los números enteros entre 0 y 213 que sean divisibles por 13.



Ejercicio

Una mañana ponés un billete en la vereda al lado del obelisco porteño. A partir de ahí, cada día vas y duplicás la cantidad de billetes, apilándolos prolijamente. ¿Cuánto tiempo pasa antes de que la pila de billetes sea más alta que el obelisco?

Datos: espesor del billete: 0.11 mm, altura obelisco: 67.5 m.



Ciclo for

Para utilizar el `for` vamos a considerar un iterador en la guarda. Las instrucciones dentro del ciclo `for` se ejecutarán en cada elemento generado por el iterador.

```
for i in range(5):  
    print(i)
```



guarda del for
donde figura el alcance

instrucciones a ejecutar en
cada paso

range()

```
for i in range(5):  
    print(i)
```

```
for i in range(0,5,1):  
    print(i)
```

```
for i in range(-5,-1):  
    print(i)
```

inicio, fin, paso

*valores por omisión:
inicio 0
paso 1*

OJO
Fin significa
*hasta antes
que ese
valor*

while vs. for

```
i = 0
```

```
while i < 5:
```

```
    print(i)
```

```
    i += 1
```

```
for i in range(0,5,1):
```

```
    print(i)
```

while vs. for

```
i = 0
```

```
while i < 5:
```

```
    print(i)
```

```
    i += 1
```

```
for i in range(0, 5, 1):
```

```
    print(i)
```

Ejercicio

Usá una iteración sobre el string cadena para agregar la sílaba 'pa', 'pe', 'pi', 'po', o 'pu' según corresponda luego de cada vocal.

```
cadena = 'Geringoso'
```

```
capadepenapa = ''
```

```
for c in cadena:
```

```
    COMPLETAR
```

```
print(capadepenapa)      # Geperipingoposopo
```

Luego hazlo con un while en vez del for.



Ejercicio

Una pelota de goma es arrojada desde una altura de 100 metros y cada vez que toca el piso salta $\frac{3}{5}$ de la altura desde la que cayó. Escribí un programa que imprima una tabla mostrando las alturas que alcanza en cada uno de sus primeros diez rebotes.

Tu programa debería generar una tabla que se parezca a esta:

```
1 60.0
2 36.0
3 21.6
4 12.96
5 7.776
6 4.6656
7 2.7994
8 1.6796
9 1.0078
10 0.6047
```



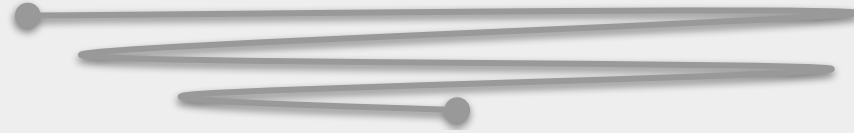
Ejercicio

Queremos hacer un traductor que cambie las palabras masculinas de una frase por su versión neutra. Como primera aproximación, completá el siguiente código para reemplazar todas las letras 'o' que figuren en el último o anteúltimo carácter de cada palabra por una 'e'. Por ejemplo 'todos somos programadores' pasaría a ser 'todes somes programadores'.

```
>>> frase = 'todos somos programadores'
>>> palabras = frase.split()
>>> for palabra in palabras:
    if ?
        ...
        frase_t = ?
        print(frase_t)
'todes somes programadores'
```



Cierre



1. Python
2. Pythontutor
3. Tipos de datos básicos + operaciones
4. Tipos de datos compuestos + operaciones
5. Condicional y ciclos